From Policy Gradient to Actor-Critic methods

The policy gradient derivation (2/3)

Olivier Sigaud

Sorbonne Université
http://people.isir.upmc.fr/sigaud

Limits of Algorithm 1

- Needs a large batch of trajectories or suffers from large variance
- The sum of rewards is not much informative
- Computing R from complete trajectories is not the best we can do

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \sim \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{H} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t^{(i)}|\mathbf{s}_t^{(i)}) R(\tau^{(i)})$$

$$\sim \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{H} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t^{(i)}|\mathbf{s}_t^{(i)}) [\sum_{t=1}^{H} r(\mathbf{s}_t^{(i)}, \mathbf{a}_t^{(i)})]$$

* split into two parts

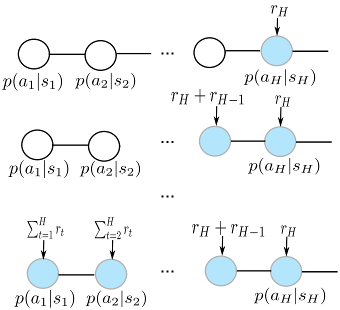
$$\sim \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{H} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t^{(i)}|\mathbf{s}_t^{(i)}) [\sum_{k=1}^{t-1} r(\mathbf{s}_k^{(i)}, \mathbf{a}_k^{(i)}) + \sum_{k=t}^{H} r(\mathbf{s}_k^{(i)}, \mathbf{a}_k^{(i)})]$$

* past rewards do not depend on the current action

$$\sim \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{H} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t^{(i)}|\mathbf{s}_t^{(i)}) [\sum_{k=t}^{H} r(\mathbf{s}_k^{(i)}, \mathbf{a}_k^{(i)})]$$

https://www.youtube.com/watch?v=S_gwYj1Q-44 (28')

Algorithm 2



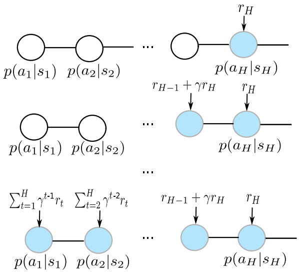
- Same as Algorithm 1
- But the sum is incomplete, and computed backwards
- Slightly less variance, because it ignores irrelevant rewards

Discounting rewards

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \sim \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{H}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\mathbf{a}_t^{(i)}|\mathbf{s}_t^{(i)})[\sum_{k=t}^{H} r(\mathbf{s}_k^{(i)},\mathbf{a}_k^{(i)})]$$

* reduce the variance by discounting the rewards along the trajectory

$$\sim \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{H}\nabla_{\boldsymbol{\theta}}\log\pi_{\boldsymbol{\theta}}(\mathbf{a}_t^{(i)}|\mathbf{s}_t^{(i)})[\sum_{k=t}^{H} \gamma^{k-t} r(\mathbf{s}_k^{(i)},\mathbf{a}_k^{(i)})]$$



https://www.youtube.com/watch?v=S_gwYj1Q-44 (39')

Introducing the action-value function

- $\sum_{k=t}^{H} \gamma^{k-t} r(\mathbf{s}_k^{(i)}, \mathbf{a}_k^{(i)})$ can be rewritten $Q_{(i)}^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t^{(i)}, \mathbf{a}_t^{(i)})$
- $$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \sim \frac{1}{m} \sum_{i=1}^{m} \sum_{t=1}^{H} \nabla_{\boldsymbol{\theta}} \log \pi_{\boldsymbol{\theta}}(\mathbf{a}_t^{(i)} | \mathbf{s}_t^{(i)}) Q_{(i)}^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t^{(i)}, \mathbf{a}_t^{(i)})$$
- It is just rewriting, not a new algorithm
- But suggests that the gradient could be just a function of the local step if we could estimate $Q_{(i)}^{\pi_{\boldsymbol{\theta}}}(\mathbf{s}_t, \mathbf{a}_t)$ in one step

Estimating $Q^{\pi_{\boldsymbol{\theta}}}(s,a)$

- Instead of estimating $Q^{\pi_{\boldsymbol{\theta}}}(s,a) = \mathbb{E}_{(i)}[Q^{\pi_{\boldsymbol{\theta}}}_{(i)}(s,a)]$ from Monte Carlo,
- Build a model $\hat{Q}^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}}$ of $Q^{\pi_{\boldsymbol{\theta}}}$ through function approximation
- Two approaches:
  - Monte Carlo estimate: Regression against empirical return

$$\boldsymbol{\phi}_{j+1} \to arg\min_{\boldsymbol{\phi}_j} \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{H}(\sum_{k=t}^{H}\gamma^{k-t}r(\mathbf{s}_k^{(i)},\mathbf{a}_k^{(i)}) - \hat{Q}^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}_j}(\mathbf{s}_t^{(i)},\mathbf{a}_t^{(i)}))^2$$

  - Temporal Difference estimate: init $\hat{Q}^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}_0}$ and fit using $(\mathbf{s},\mathbf{a},r,\mathbf{s}')$ data

$$\boldsymbol{\phi}_{j+1} \to \min_{\boldsymbol{\phi}_j} \sum_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')} ||r + \gamma f(\hat{Q}^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}_j}(\mathbf{s}',.)) - \hat{Q}^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}_j}(\mathbf{s},\mathbf{a})||^2$$

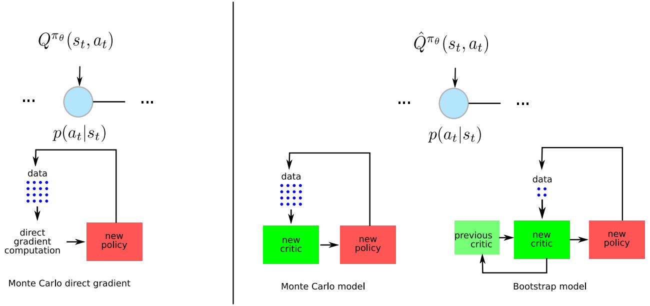
  - $f(\hat{Q}^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}_j}(\mathbf{s}',.)) = \max_{\mathbf{a}} \hat{Q}^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}_j}(\mathbf{s}',\mathbf{a})$ (Q-learning), $= \hat{Q}^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}_j}(\mathbf{s}',\pi_{\boldsymbol{\theta}}(\mathbf{s}'))$ (AC)...
  - May need some regularization to prevent large steps in ϕ

https://www.youtube.com/watch?v=S_gwYj1Q-44 (36')

Martin Riedmiller. Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pp. 317–328. Springer, 2005

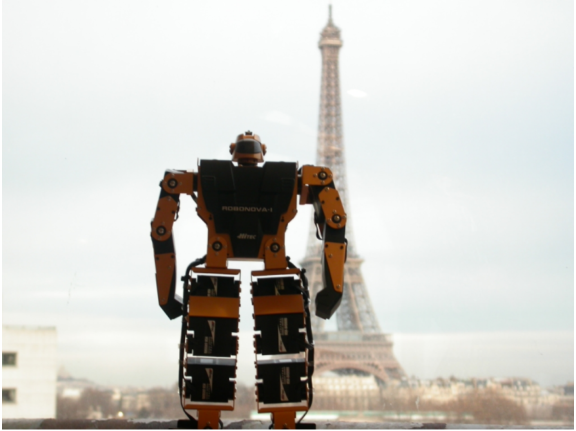
András Antos, Csaba Szepesvári, and Rémi Munos. Fitted Q-iteration in continuous action-space MDPs. In *Advances in neural information processing systems*, pp.9–16, 2008.

Monte Carlo versus Bootstrap approaches



- Three options:
  - MC direct gradient: Compute the true $Q^{\pi_{\boldsymbol{\theta}}}$ over each trajectory
  - MC model: Compute a model $\hat{Q}^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}}$ over rollouts using MC regression, throw it away after each policy gradient step
  - Bootstrap: Update a model $\hat{Q}^{\pi_{\boldsymbol{\theta}}}_{\boldsymbol{\phi}}$ over samples using TD methods, keep it over policy gradient steps
- With bootstrap, update everything from the current state, see next lessons
- Next lesson: adding a baseline

Any question?



Send mail to: `Olivier.Sigaud@upmc.fr`

András Antos, Csaba Szepesvári, and Rémi Munos.
Fitted Q-iteration in continuous action-space mdps.
In *Advances in neural information processing systems*, pp. 9–16, 2008.

Martin Riedmiller.
Neural fitted Q iteration–first experiences with a data efficient neural reinforcement learning method.
In *European Conference on Machine Learning*, pp. 317–328. Springer, 2005.